WHAT IS CLAIMED IS:

1. A real-time OS simulator that assigns a task processing thread to run on a general-purpose multi-thread OS to each of a plurality of tasks to run on a real-time OS and simulates an operation of said real-time OS on said multi-thread OS, said simulator comprising:

task switching instruction means for receiving a request issued from said task processing thread under same conditions as in said real-time OS, and providing an instruction for switching the tasks in response to said request; and

a task switching thread for making selected one of said task processing threads run by suspending and resuming said task processing threads with capabilities of said multi-thread OS in cooperation with said task switching instruction means.

2. The real-time OS simulator according to claim 1, wherein

said task switching instruction means selects a task processing thread to run next, provides the instruction for switching the tasks to said task switching thread, and then suspends the task processing thread that has issued said request, and

in response to the instruction, said task switching thread resumes the selected task processing thread after a preceding running task processing thread is suspended.

3. The real-time OS simulator according to claim 2, wherein

in response to the instruction for switching the tasks, said task switching thread checks at predetermined intervals whether the preceding running task thread is suspended or not.

4. The real-time OS simulator according to claim 1, wherein

said task switching instruction means selects a task processing thread to run next, provides the instruction for switching the tasks to said task switching thread, and then sets the task processing thread that has issued said request in a waiting state, and

in response to the instruction, said task switching thread suspends a preceding running task processing thread, and then releases the selected task processing thread from the waiting state for resuming.

5. The real-time OS simulator according to claim 2 or 4, wherein

said task switching instruction means provides the instruction to said task switching thread after said task switching thread is enabled to start processing.

6. The real-time OS simulator according to claim 1, wherein

said task switching instruction means provides the

instruction to said task switching thread after selecting a task processing thread to run next, and

said task switching thread runs with a higher priority than said task processing threads and, in response to the instruction, suspends a preceding running task processing thread and then resumes the selected task processing thread.

7. The real-time OS simulator according to claim 1, further comprising

task processing thread creating means for creating said task processing thread.

8. The real-time OS simulator according to claim 1, wherein

an exception handling thread corresponding to task exception handling of each of said tasks and running on said multi-thread OS is further assigned to each of said tasks, and

said task switching thread selects a thread to run next from among said task processing threads and said exception handling threads.

9. The real-time OS simulator according to claim 8, further comprising

thread creating means for creating said task processing thread and said exception handling thread.

10. The real-time OS simulator according to claim 1, further comprising

interrupt handling means for receiving an interrupt request issued by an interrupt thread that generates a pseudo-interrupt, suspending a running task processing thread, calling an interrupt handler corresponding to the interrupt request, and then selecting a task processing thread to run next for resuming.

11. The real-time OS simulator according to claim 10, wherein

when receiving the interrupt request from said interrupt thread while another interrupt thread is running, said interrupt handling means suspends the running interrupt thread, calls the interrupt handler corresponding to the interrupt request, and then resumes the suspended interrupt thread.

12. The real-time OS simulator according to claim 10, wherein

said interrupt thread includes a system clock interrupt thread that generates a pseudo-interrupt at predetermined time intervals.

13. The real-time OS simulator according to claim 10, further comprising

interrupt thread creating means for creating said interrupt thread.

14. A computer-readable recording medium recording a program to run on a computer, the program for a simulation method of assigning a task processing thread to run on a general-purpose multi-thread OS to each of a plurality of tasks to run on a real-time OS and simulating an operation of said real-time OS on said multi-thread OS, said simulation method comprising the steps of:

receiving a request issued from said task processing thread under same conditions as said real-time OS and providing an instruction for switching the tasks in response to said request; and

making selected one of said task processing threads run by suspending and resuming said task processing threads with capabilities of said multi-thread OS.

15. The recording medium according to claim 14, wherein

in said instruction providing step, a task processing thread to run next is selected, and then the task processing thread that has issued said request is suspended, and

in said run step, the selected task processing thread is resumed after a preceding running thread is suspended.

16. The recording medium according to claim 14, wherein

in said instruction providing step, a task processing thread to run next is selected, and then the task processing thread that has issued said request is set to a waiting state, and

in said run step, after a preceding running task processing thread is suspended, a waiting state of the selected task processing thread is cleared for resuming.

17. The recording medium according to claim 14, wherein

in said instruction providing step, a task processing thread to run next is selected, and

said run step is given a higher priority than said task processing threads, and, in said run step, after a preceding running task processing thread is suspended, the selected task processing thread is resumed.

18. The recording medium according to claim 14, wherein

an exception handling thread corresponding to task exception handling of each of said tasks and running on said multi-thread OS is further assigned to each of said tasks, and

in said run step, a thread to run next is selected from among said task processing threads and said exception handling thread.

19. The recording medium according to claim 14, wherein

said simulation method further comprises the step of receiving an interrupt request issued from an interrupt thread that generates a pseudo-interrupt, suspending a running task processing thread, calling an interrupt handler corresponding to the interrupt request, and then selecting a task processing thread to run next for resuming.

20. A program for a simulation method of assigning a task processing thread to run on a general-purpose multi-thread OS to each of a plurality of task to run on a real-time OS and simulating an operation of said real-time OS on said multi-thread OS, said simulation method comprising the steps of:

receiving a request issued from said task processing thread under same conditions as said real-time OS and providing an instruction for switching the tasks in response to said request; and

making selected one of said task processing threads run by suspending and resuming said task processing threads with capabilities of said multi-thread OS.

21. The program according to claim 20, wherein

in said instruction providing step, a task processing thread to run next is selected, and then the task processing thread that has issued said request is suspended, and

in said run step, the selected task processing thread

is resumed after a preceding running thread is suspended.

22. The program according to claim 20, wherein

in said instruction providing step, a task processing thread to run next is selected, and then the task processing thread that has issued said request is set to a waiting state, and

in said run step, after a preceding running task processing thread is suspended, a waiting state of the selected task processing thread is cleared for resuming.

23. The program according to claim 20, wherein

in said instruction providing step, a task processing thread to run next is selected, and

said run step is given a higher priority than said task processing threads, and, in said run step, after a preceding running task processing thread is suspended, the selected task processing thread is resumed.

24. The program according to claim 20, wherein

an exception handling thread corresponding to task exception handling of each of said tasks and running on said multi-thread OS is further assigned to each of said tasks, and

in said run step, a thread to run next is selected from among said task processing threads and said exception handling thread.

25. The program according to claim 20, wherein said simulation method further comprises the step of receiving an interrupt request issued from an interrupt thread that generates a pseudo-interrupt, suspending a running task processing thread, calling an interrupt handler corresponding to the interrupt request, and then selecting a task processing thread to run next for resuming.